多値Signed-Digit数演算VLSIアレー(多値論理及びその応用(4))

AUTHOR(S):

川人, 祥二; 亀山, 充隆; 樋口, 龍雄

数理解析研究所講究録
687巻 1989年 233-248

多値Signed-Digit数演算VLSIアレー

東北大学　工学部
川人祥二 (Shoji Kawahito)
亀山充隆 (Michitaka Kameyama)
樋口龍雄 (Tatsuo Higuchi)

1. まえがき

1チップ上に10^8個以上の素子を集積したVLSIやWafer Scale Integration(WSI)等の超大規模かつ超高速の集積回路を実現する上では、設計及びテストが容易であり、高並列性を有するVLSIアーキテクチャが極めて重要なものとなり、最近Systolic Arrayに代表されるモジュラアレー構造のVLSIアーキテクチャが注目されてきている[1]。Systolic Arrayは、徹底的にパイプライン処理を駆使することにより、スループットの向上を目指したものであるが、知能ロボットの実時間制御等の特定の応用に対しては、スループットの向上よりも個々の処理回路の絶対的な演算遅れ時間を最小化することが重要であり、Systolic Arrayなどとは異質のVLSIアーキテクチャが要求される[2]。

演算遅れ時間を小さくする上では、Signed-Digit数系（以下SD数系）等の特殊な数表現が有効である[3]。SD数表現を利用した集積回路としては、これまで入出力を2進数表現とし、内部にSD数表現を利用した乗算器等のLSIが幾つか試作され、その有効性

が明らかにされてきている[4,5]。大規模なＶＬＳＩシステムの中で、ＳＤ数表現の演算遅れ時間の小さい特長をより明確にするためには、入出力をもＳＤ数表現とする演算方式が有効であるが、ＳＤ数表現が冗長表現であるために回路構成および配線が非常に複雑になる。そこで本稿ではまず、回路構成が大幅に簡単化され、より高速化の期待できる拡張４進ＳＤ数表現に基づく演算アルゴリズムを提案する。このような拡張４進ＳＤ数演算回路は、多値双方向電流モード回路方式により効率よく構成でき、能動素子数、配線数が少ないコンパクトに実現が可能となる[6]。

さらに、このような拡張４進ＳＤ数表現に基づく演算回路の規則性に着目し、絶対的な演算遅れ時間が小さい算術演算システムを、モジュラアレー構造で実現できるＶＬＳＩアーキテクチャを提案する。まず、プロダクト発生回路、並列加算器及びその他の簡単な制御回路を内蔵するモジュールを基本モジュールとして定義する。これをアレー状に配置し、モジュール間の相互配線を施すと共に、モジュールの機能を選択するための簡単な配線によるマスクプログラムを行うことにより、行列演算等の様々な算術演算ＶＬＳＩシステムが実現できる。このようなＶＬＳＩアレーは、設計及びテストが容易であり、絶対遅れ時間が極めて小さいという性質を有し、例えば16ビット精度の４×４の行列乗算が、全加算器５数段程度の遅れで実行できる。

２．拡張４進ＳＤ数演算アルゴリズム

2．1　定義と並列加算アルゴリズム

拡張4進SD数表現は、n桁の数

$$X=(x_{n-1}\cdots x_i\cdots x_0)_{MSD4}=\sum_{i=0}^{n-1}x_i 4^i \qquad (1)$$

において、各桁x_iが、次式の値をとる数表現であると定義する。

$$x_i \in L = \{-2,-1,0,1,2\} \qquad (2)$$

これは冗長表現であるが、通常の4進SD数表現が{-3,-2,-1,0,1,2,3}の7値を用いるのに対して、拡張4進SD数表現の方がより冗長性の小さい表現となっている。拡張4進SD数系における並列加算は、各桁において次式のような3つの手続きにより実行される。

$$z_i = x_i + y_i \qquad (3)$$

$$4c_i + w_i = z_i \qquad (4)$$

$$s_i = w_i + c_{i-1} \qquad (5)$$

ここで、$z_i \in \{-4,\ldots,0,\ldots,4\}$は、入力線形和、$w_i \in \{-2,-1,0,1,2\}$は、中間和、$c_i \in \{-1,0,1\}$は、桁上げであり、$s_i \in \{-2,-1,0,1,2\}$は、最終和である。式(4)は、$z_i$から$w_i$, c_iを求める演算であるが、$s_i \in L$とするために、$z_i = \pm 2$のときには、下位からの桁上がりを考慮して、z_{i-1}も同時に用いて次式のように決定する。

$$\begin{cases} w_i = z_i - 4,\ c_i = 1 & (z_i \geqq 3) \\ w_i = z_i - 4,\ c_i = 1 & (z_i = 2,\ z_{i-1} \geqq 0) \\ w_i = z_i,\ \ c_i = 0 & (z_i = 2,\ z_{i-1} < 0) \\ w_i = z_i,\ \ c_i = 0 & (-1 \leqq z_i \leqq 1) \\ w_i = z_i,\ \ c_i = 0 & (z_i = -2,\ z_{i-1} \geqq 0) \\ w_i = z_i + 4,\ c_i = -1 & (z_i = -2,\ z_{i-1} < 0) \\ w_i = z_i + 4,\ c_i = -1 & (z_i \leqq -3) \end{cases} \qquad (6)$$

このような並列加算アルゴリズムでは、キャリc_iは、c_{i-1}に依存

しないため、語長に無関係な一定時間で、並列加算を実行することができ、データの語長の長い演算を非常に高速に実行できる。なお、拡張4進SD数表現による加算はは、並列性の点では通常の4進SD数表現の場合と同様であるが、回路構成上は有利な性質を有しており、より高速な演算回路が実現できる。

2．2　拡張4進SD数乗算アルゴリズム

並列乗算器の入出力及び内部表現として、拡張4進SD数表現を利用すれば、非常に並列性の高いものとすることができ、さらに回路構成上も部分積生成回路が非常に簡単化できるなど興味深い性質が見いだせる。

比較のために、まず入出力を通常の4進SD数表現とする並列乗算器について考える。この場合、SD数系が冗長表現であるために部分積生成が複雑になるという問題点があり、コンパクトな実現が難しい。4進SD数表現の被乗数X，乗数Yの各桁は、$L'=\{-3,-2,-1,0,1,2,3\}$の値を取る。　従って、各桁毎の積すなわち部分積$q_{i,j}=x_i y_j$は、次のような値となる。

$$q_{i,j}\in\{-9,-6,-4,-3,-2,-1,0,1,2,3,4,6,9\} \qquad (7)$$

さらに部分積を4進SD数加算器を用いて加算するためには、このような部分積をL'の値をとるように変換する必要がある。このような変換まで含めた部分積生成回路は、非常に複雑なものとなる。

これに対して、拡張4進SD数表現を用いれば、部分積の生成回路が極めて簡単化される。　なぜなら、乗数Y，被乗数Xの、各桁y_j，x_iは$L=\{-2,-1,0,1,2\}=\{-2^1,-2^0,0,2^0,2^1\}$の値を取るので、部分積生成をシフト操作と符号反転の操作により行うことができるからである[7]。　まず、シフト操作が行えるよう、拡張4進SD数

表現の被乗数Xの各桁x_iを次式に基づき分解し、2進SD数表現に変換する。

$$2a_i + b_i = x_i \tag{8}$$

ここで、a_i, $b_i \in \{-1, 0, 1\}$である。このとき、部分積$p_{i,j}$は、y_jの値に対応して、次式のように求められる。

$$p_{i,j} = \begin{cases} 2b_i + a_{i-1} & (y_j = 2) \\ 2a_i + b_i & (y_j = 1) \\ 0 & (y_j = 0) \\ -2a_i - b_i & (y_j = -1) \\ -2b_i - a_{i-1} & (y_j = -2) \end{cases} \tag{9}$$

このようにして生成された部分積が、$p_{i,j} \in L$となるならば、部分積オペランド$P_j = (p_{n,j} \cdots p_{i,j} \cdots p_{0,j})_{MSD4}$は、拡張4進SD数表現であり、後はこれらのオペランドを拡張4進SD数並列加算器を用いて、多入力加算を行えば最終積が求められる。しかしながら、

$$b_i = a_{i-1} = 1, \quad b_i = a_{i-1} = -1 \tag{10}$$

のどちらかの場合$p_{i,j} = \pm 3$となり、$p_{i,j} \in L$なる条件が満たされない。そこで、式(10)の値とならないよう、式(8)に基づきa_i, b_iを決定する際に、x_{i-1}の値も同時に用いて行う。この手法については、3章で説明する。

部分積オペランドの多入力加算は拡張4進SD数並列加算器を2進木構造に接続して行うことにより、非常に高速なものとすることができる。乗数nビット、被乗数mビットの乗算器の場合、次式の全加算器通過段数ℓで、最終的な積が求められる。

$$\ell = \lfloor \log_2 (n / 2) \rfloor \tag{11}$$

ここで、⌊ｘ⌋は、⌊ｘ⌋≧ｘなる最小の整数である。例えば、３２×３２ビットの精度では、１段の部分積生成回路と４段の全加算器の遅延で、最終積が得られる。なお、２進数入出力の乗算器との最も大きな違いは、乗算時間が被乗数の語長に無関係であることであり、このような意味からも、特に長い語長の場合に有利な性質を有している。

２．３　拡張４進ＳＤ数除算アルゴリズム

いま、被除数Ｘ、除数Ｄが共に、拡張４進ＳＤ数表現であり、拡張４進ＳＤ数表現の商$Q=(q_0.q_1q_2\cdots q_j\cdots q_n)_{MSD4}$, $q_j\in L$を求める除算器を考える。部分剰余を$R^{(j)}$とすると、除算アルゴリズムは、次式の漸化式で表される。

$$R^{(j+1)}=4R^{(j)}-q_{j+1}D \qquad (12)$$

ここで、$R^{(0)}=X$である。いま、Ｄの最上位桁が０でない値を取るよう正規化されており、$X\leqq 2D/3$が成り立っているとすると、部分剰余$R^{(j+1)}$及び商ディジットq_{j+1}を求める演算は、次式のように表される。

$$R^{(j+1)}=\begin{cases}4R^{(j)}+2D\\ 4R^{(j)}+D\\ 0\\ 4R^{(j)}-D\\ 4R^{(j)}-2D\end{cases}\qquad q_{j+1}=\begin{cases}-2 & (4R^{(j)}\leqq -3D/2)\\ -1 & (-3D/2<4R^{(j)}\leqq -D/2)\\ 0 & (-D/2<4R^{(j)}\leqq -D/2)\\ 1 & (D/2<4R^{(j)}\leqq 3D/2)\\ 2 & (3D/2\leqq 4R^{(j)})\end{cases}\qquad (13)$$

q_{j+1}、$4R^{(j)}$、ＤとＲ$^{(j+1)}$の関係は図１のようになる。一点鎖線は、$|R^{(j+1)}|\leqq|2D/3|$、$|4R^{(j)}|\leqq|8D/3|$を満たす範囲を示しており、商ディジットq_{j+1}はこれを満たすよう決定する。この際に、D/2，3D/2と$4R^{(j)}$との比較が必要であり、これを正確に行おうとすると

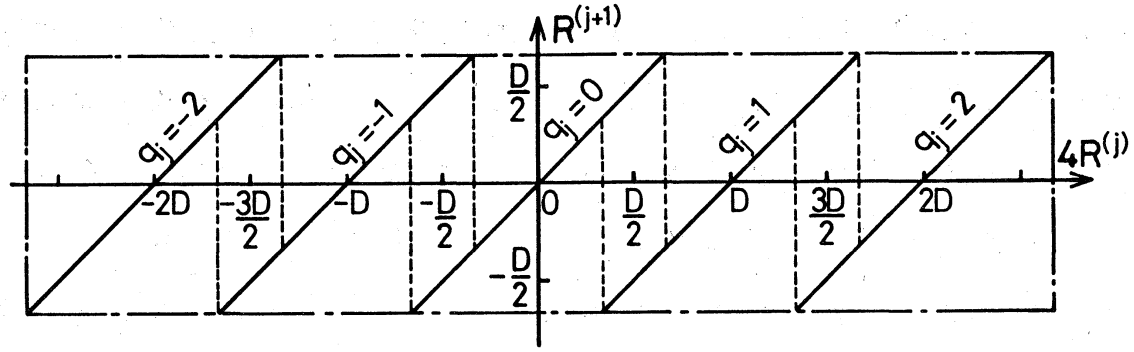


図 1　q_{j+1}, $4R^{(j)}$, Dと$R^{(j+1)}$の関係

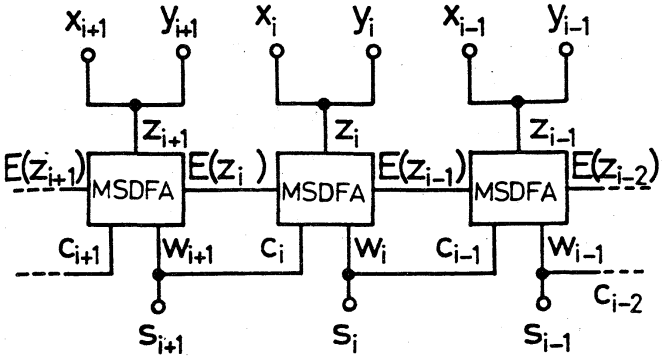


図 2　並列加算器

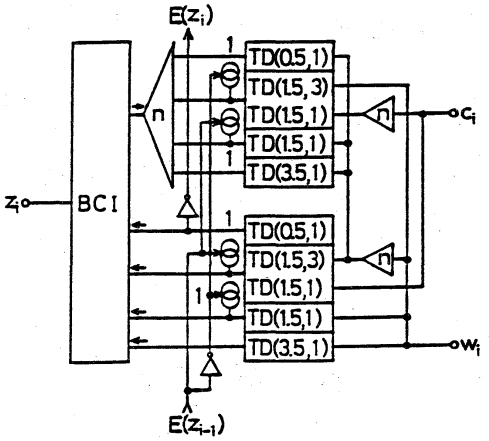


図 4　拡張ＳＤ全加算器（MSDFA）

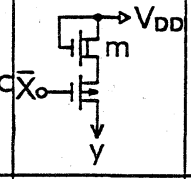
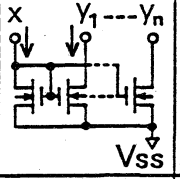
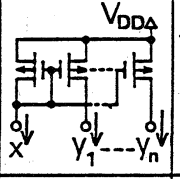
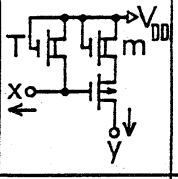
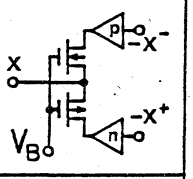
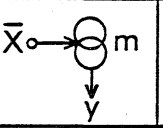
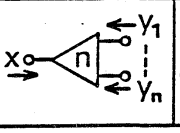
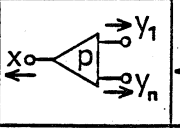
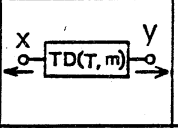
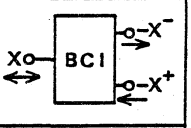
BASIC CIRCUIT	CURRENT SOURCE	CURRENT MIRRORS N-CH. TYPE	CURRENT MIRRORS P-CH. TYPE	THRESHOLD DETECTOR	BIDIRECTIONAL CURRENT INPUT
SCHEMATIC					
SYMBOL					
FUNCTION	$\begin{cases} y=0 & \text{if } \bar{X}=\text{"1"} \\ y=m & \text{if } \bar{X}=\text{"0"} \end{cases}$	$y_i=-a_i x$ for $i=1,\ldots,n$ a_i:SCALE FACTOR		$\begin{cases} y=0 & \text{if } x\le T \\ y=m & \text{if } x>T \end{cases}$	$\begin{cases} x^+=x\ x^-=0 & \text{if } x\ge 0 \\ x^+=0\ x^-=x & \text{if } x<0 \end{cases}$

図 3　双方向電流モード基本演算素子

多大な時間を要するが、拡張４進ＳＤ数表現が冗長表現であるために比較が非常に簡単化される。　例えば$|4R^{(j)}|$が$|D/2|$付近の値であり、q_{j+1}が０か１かを決定する場合、$|D/2|\pm|D/6|$の範囲ではq_{j+1}として０と１の両方が許されるため、比較の精度としてDの

1/6以下の誤差が許される。すなわちD及び$R^{(j)}$の上位4桁のみの比較を行えばよいことになる。また、$R^{(j+1)}$を求める際の加減算は、拡張4進SD数並列加算器を用いて高速に実行できる。

なお、商の表現にSD数表現を用いる除算アルゴリズムとして、SRT除算、Robertsonの高基数除算などがあるが、これらは入出力を2進数表現とするものであるので、加減算に多くの時間を要し、SD数表現から2進数表現への変換が必要となる[8]。これに対して本除算法では、拡張4進SD数表現を入出力とし、拡張4進SD数表現の商が直接得られるという意味でも非常に高速な除算器が実現できると考えられる。

3. 多値双方向電流モード演算回路

3.1 基本演算回路

拡張4進SD数演算回路は、多値双方向電流モード回路方式により非常に効率よく構成できる。これは、変数の絶対値を電流レベルで、符号を電流方向で表現して回路を構成するものであり、特に符号まで考慮して、線形加算を結線により実行できるという性質により、能動素子数、配線数を非常に少なくできる[9]。例えば並列加算器は、図2のように非常に簡単な構造で実現できる。先に述べた加算アルゴリズムの中で式(3)と(5)の演算は、符号を含む線形加算演算であり、これは単なる結線で実現できる。式(4)に対応する演算は、式(6)より明かなようにz_i, z_{i-1}からw_i, c_iを生成する演算であり、これを実行する演算回路を拡張SD全加算器（MSDFA）と呼ぶ。これは、図3の示す双方向電流モードの基本演算素

子を用いて図４のように58個のトランジスタで構成できる。図４のＭＳＤＦＡでは、各出力レベルが量子化された入出力特性が直接得られる。通常のＳＤ全加算器では、中間和出力のレベルを量子化するための量子化回路を接続する必要があったが、ＭＳＤＦＡでは、量子化回路を接続する必要がないため、より一層の高速化が期待できる。

２進数並列加算器と比較した場合、拡張４進ＳＤ数並列加算器は、長い語長の場合に格段に高速であると同時に構造も極めて簡単であり、素子数、配線数も少ないため、超大規模集積化に適した、高速性とコンパクト性を兼ね備えたものとなる。

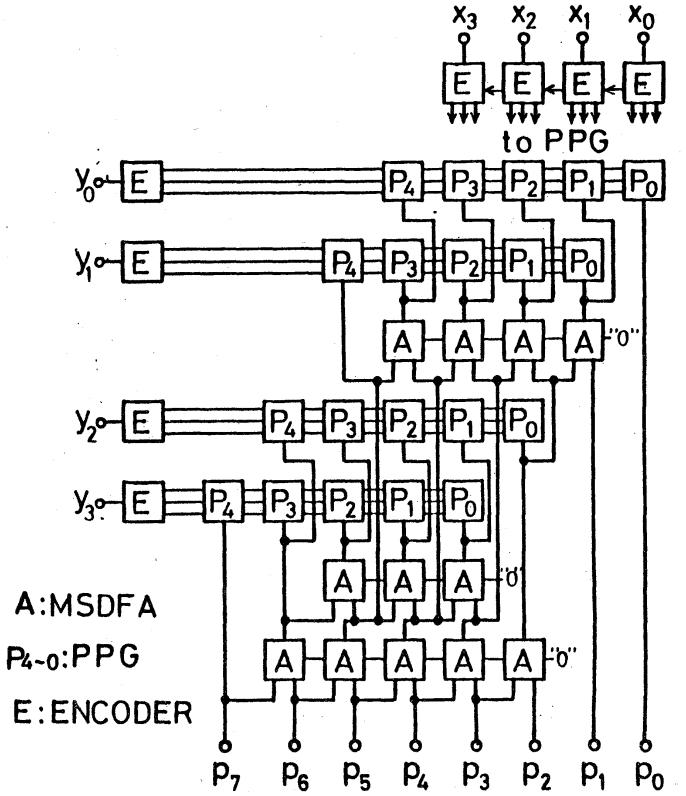


図５　４×４桁の拡張４進ＳＤ数乗算器

３．２　並列乗算器

拡張４進ＳＤ数表現を入出力とする並列乗算器は、例えば４×４桁の場合、図５のように構成でき、非常に規則的かつ簡単な構造と

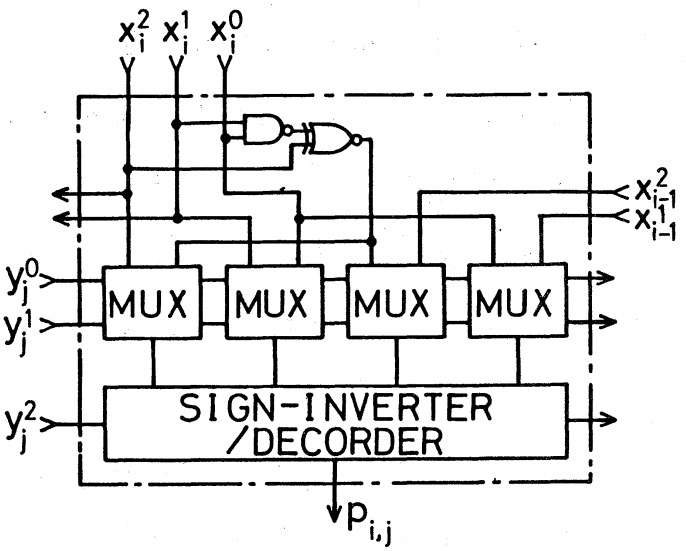


図６　部分積生成回路(PPG)

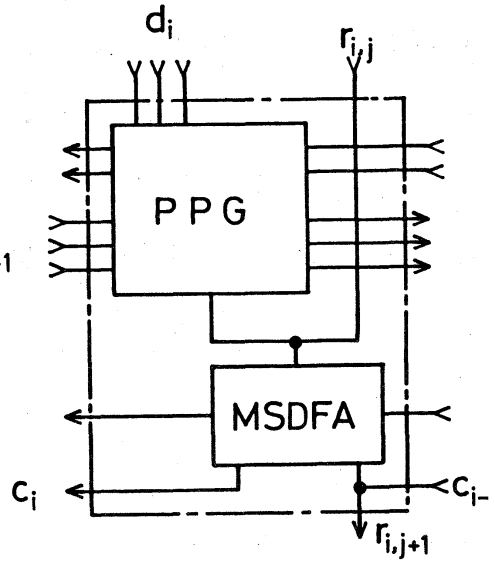


図７　除算モジュール

表１　y_jの２値符号化

y_j	$y^2{}_j$	$y^1{}_j$	$y^0{}_j$
2	1	1	0
1	1	0	1
0	0	0	0
-1	0	0	1
-2	0	1	0

表２　x_iの２値符号化

x_i	x_{i-1}	$x^2{}_i$	$x^1{}_i$	$x^0{}_i$
2	−	1	1	0
1	≧ 1	1	1	1
1	＜ 1	1	0	1
0	−	0	0	0
-1	≧ 1	0	0	1
-1	＜ 1	0	1	1
-2	−	0	1	0

なる。部分積加算は、拡張４進ＳＤ数並列加算器を用いた２進木構造により、高速に実行できる。部分積生成回路は、2.2で述べたようにシフト演算と符号反転演算を実行する回路である。まず被乗数、乗数の各桁を表１，２に従ってそれぞれ２値符号化を行う。表２において、$x_i = \pm 1$のときは、式(10)のようにならないよう$x_{i-1} \geqq 1$と$x_{i-1} < 1$の場合によって符号化を変更する。部分積生成回路(PPG)は図８のように構成され、表１，２の符号化を行うエンコーダからの信号に対してシフト演算、符号反転の演算を行い、最終段（デコーダ）において多値電流モード信号に変換し、部分積を生成する。図６の回路は、58素子で実現できる。

3．3　並列除算器

2．3で述べたアルゴリズムに従い、拡張4進SD数除算器は配列形構造により規則的かつ簡単な構造で実現できる。式(15)に対応する演算の1桁を実行するモジュールは、図7のように乗算器のPPGモジュールとMSDFAモジュールを用いて構成できる。　また、Dおよび$R^{(j)}$の上位4ビットに対しD/2及び3D/2を生成し、Dと$4R^{(j)}$の符号の比較、D/2，3D/2と$4R^{(j)}$の比較を行うことにより商ディジットq_{j+1}を生成する回路が必要である。語長が長い場合、全体に占めるこの回路のハードウェア量の割合は小さくなる。なお、この回路では、求めれた商ディジットq_{j+1}から、シフト信号（$q^1{}_{j+1}$，$q^0{}_{j+1}$）、符号反転制御信号$q^2{}_{j+1}$を生成する。

4．SD数演算VLSIアレー

ここでは、拡張4進SD数演算をより大規模な処理回路で利用し、絶対的な演算遅れ時間が小さく、単一モジュールの規則的な配列で実現できるVLSIアレーについて考える。

4．1　基本モジュールの定義

図8に示す回路を基本モジュールとして定義する。これをPAM（Productgenerator & Adder Module）モジュールと呼ぶことにする。拡張4進SD数演算回路は、構造的な規則性があり、また図7のように乗算器の部分積生成回路と全く同じ回路を除算器にも用いることができるという性質がある。このような性質により、これを繰り返し用いることにより、様々な算術演算回路が構成できる。PAMモジュールは、次式のような3種類の演算を行う。

$$① S = R + y X \qquad (14a)$$

$$② \begin{cases} S = R + U \\ V = y X \end{cases} \qquad (14b)$$

$$③ \quad S = R - q X \qquad (14c)$$

ここで、S，R，U，V，Xは、拡張4進SD数表現のオペランドであり、$y, q \in \{-2, -1, 0, 1, 2\}$である。①と②は、乗算器を2進木構造で実現するために必要であり、③は、除算器を構成する場合の演算である。SELECTOR及びSEは、モジュールの配置後、配線によるマスクプログラムを行うことによって①，②及び③の機能のどれかを選択するためのものである。

拡張SD数演算回路ではまた、演算の高並列性を保存したまま、ディジットスライス接続が可能であるという性質がある。PAMモジュールにおいても、図8の一点鎖線上の端子間の接続により、語長nのモジュールをm個接続して語長n×mのモジュールが容易に構成できる。なおこのためSEにディジットスライス接続の機能を選択できるようにしている。

このようなモジュールを加減算のために使用する場合、①の機能を選択することにより、(1)S＝R＋X，(2)S＝R＋2X，(3)S＝R－X，(4)S＝R－2Xの4種類の演算が行える。

語長8桁のPAMモジュールを用いて、8×8桁の乗算器、除数8桁、被除数8桁の除算器が、図9のように構成できる。数100ビットの語長の超高速の乗算器、除算器の構成においては、このようなモジュラアレー構造による実現が特に有効であると考えられる。

4．2　内積演算回路，行列演算への応用

4．1で定義した基本モジュールを用いて、乗算器，除算器だけではなく、内積演算、行列演算回路なども規則的な単一モジュール

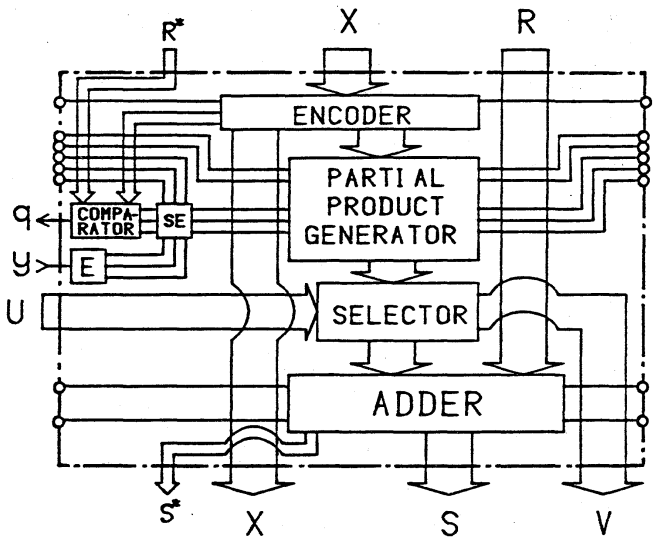
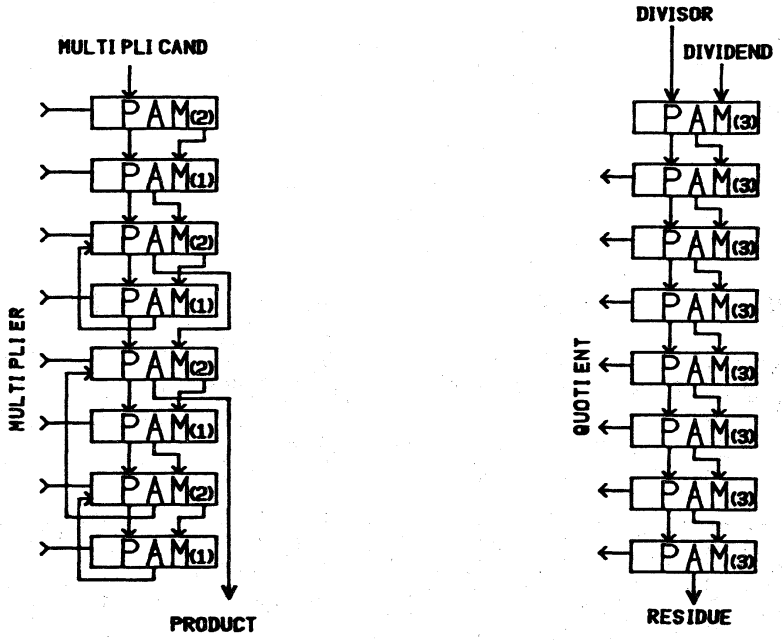


図８　ＰＡＭモジュール



（ａ）乗算器　　（ｂ）除算器

図９　乗算器，除算器の構成

の配列構造で実現できる。例えば、次式で与えられる内積演算を考える。

$$c = a_1 b_1 + a_2 b_2 + a_3 b_3 + a_4 b_4 \qquad (15)$$

このような演算を４桁の語長で並列に実行する回路は、４桁のＰＡＭモジュールを用いて、図10のような構造で実現できる。構造は

全体として2進木構造となっており、この場合、わずか1段の部分積生成回路と5段の全加算器の遅延で最終結果が得られ、極めて高速な演算システムが構築できる。このような構造により、一般には、語長nビット、m項の内積演算を次式に示す全加算器段数ℓで、実行できる。

$$\ell = \lfloor \log_2 n + \log_2 m - 1 \rfloor \qquad (16)$$

また、このような演算システムを単位として、次式で与えられる4行4列の行列乗算システムを、図11のように構成できる。

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix} \qquad (17)$$

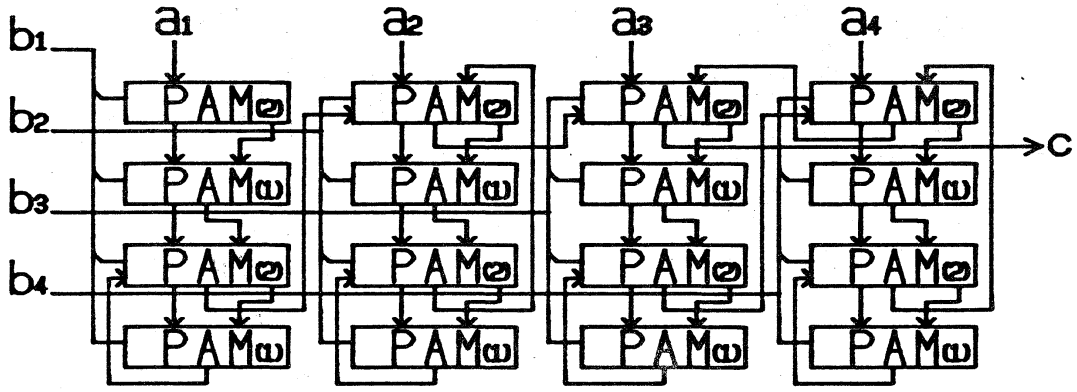


図10　内積演算回路（ＩＰ）

このように規則的な構造でＶＬＳＩとして実現でき、各出力の演算遅れ時間は、全加算器5段程度であり、極めて高速な演算が行える。状態フィードバック、観測器を用いたディジタル制御システム等においては、多段にわたる複雑な行列演算、特に行列乗算を直列的に実行する必要があり高速な実時間制御は容易ではないが、演算遅れ

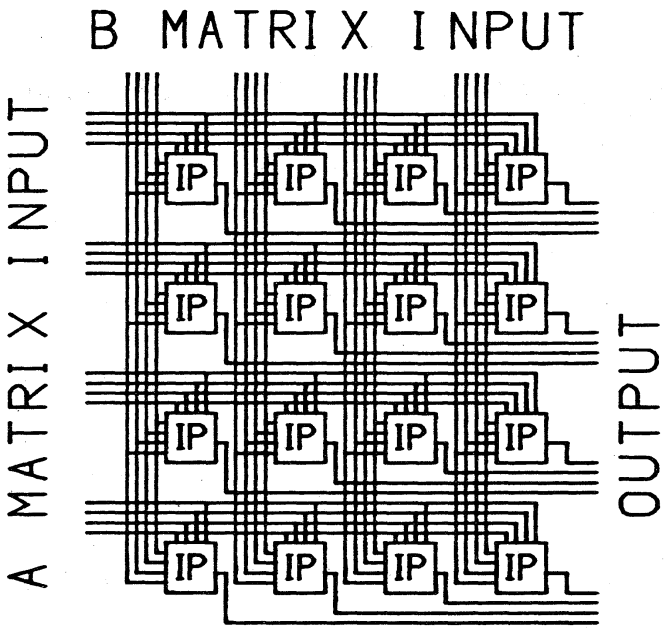


図11　4×4の行列乗算回路

時間の小さい本ＶＬＳＩアレーが、有効に利用できると考えられる。

５．むすび

本稿では、演算遅れ時間が小さく、構造の規則性に優れ、ハードウェア量の少ない多値双方向電流モードに基づく拡張４進ＳＤ数演算回路を提案した。さらに、その構造の規則性に着目し、単一モジュールの規則的な配列で、行列演算等の様々な算術演算システムが実現できるＶＬＳＩ向きのアーキテクチャを提案した。

なお、除算器を含んだ演算システムを本ＶＬＳＩアレーで実現する場合、正規化を行う必要があり、この方法について今後検討したい。また、より広い演算システムへの応用を考えるとき、本稿で提案したモジュールよりも汎用性があり、効率のよいモジュールも考えられる。

文 献

(1)Special issue on Systolic Arrays, IEEE Computer, 21, 7(1987)

(2)亀山, 樋口:"ロボットとＶＬＳＩコンピュータ", ロボット学会誌, Vol. 6, No. 4, pp. 64-70(昭63-8).

(3)S. Kawahito, M. Kameyama, T. Higuchi and H. Yamada:"A 32 x 32 Bit Multiplier Using Multiple-Valued MOS Current-Mode Circuits", IEEE J. Solid-State Circuits, SC-23, No. 1, pp. 124-132(Feb. 1988).

(4)H. Edamatsu et al.:"A 33 MFLOPS Floating-Point Processor Using Redundent Binary Representation", Dig. ISSCC88, pp. 152-153(1988).

(5)A. Avizienis:"Signed-Digit Number Representations for Fast Parallel Arithmetic", IRE Trans. Elect. Comput., EC-10, pp. 389-400(Sept. 1987).

(6)川人, 亀山, 樋口:"ＶＬＳＩ向き４進Signed-Digit数多値演算回路の構成", 信学論(D), J69-D, 5, pp. 679-689(昭和61-05).

(7)S. Kawahito, M. Kameyama, T. Higuchi and H. Yamada:"A High-Speed Compact Multiplier Based on Multiple-Valued Bi-Directional Current-Mode Circuits", Proc. 1987 ISMVL, pp. 172-180(May 1987).

(8)K. Hwang:"Computer Arithmetic", John Wiley & Sons, New York (1976).

(9)亀山, 川人, 樋口:"Signed-Digit数系に基づく双方向電流モード多値基本演算回路とその評価", 信学論(D), Vol. J71-D, No. 7, pp. 1189-1198(昭63-7).